Combining Information State Update, Harel Statecharts and LLMs for controllable and flexible Conversational AI

Vladislav Maraev[*,†] and **Alexander Berman**[*] and **Staffan Larsson**[*,†]
[*]Department of Philosophy, Linguistics and Theory of Science
University of Gothenburg, Sweden
[†]Talkamatic AB, Sweden
Correspondence: vladislav.maraev@gu.se

Abstract

The rise of LLM-based approaches to dialogue systems has created an increased need for controllable dialogue. This paper addresses this need by presenting an implementation of a dialogue system based on information state update approach according to Larsson (2002). This enables the integration of rule-based handling of dialogue, expressed by Harel's statecharts (1987), and Larsson's theoretical account grounded in theories of dialogue, expressed by information state update rules. We demonstrate how our approach applies to dialogue domains involving form-filling. We also propose how LLMs can be employed to inject domain knowledge and be used in various components of a hybrid dialogue system, while maintaining control over the overall dialogue logic.

1 Introduction

Despite considerable efforts to control large language models (LLMs), risks of hallucinations and related phenomena have yet to be eliminated (Xu et al., 2024; Ayyamperumal and Ge, 2024). Still, the otherwise impressive capabilities of LLMs have raised the bar for conversational AI. This makes a compelling argument for finding ways to complement LLMs with rule-based approaches that can mitigate the risks associated with using LLMs.

In this work we use the influential Information State Update (ISU) framework (Larsson and Traum, 2000; Larsson, 2002). The basis of this framework is a representation of the dialogue context as data structure which includes the information available to each participant of the dialogue (either a human or an artificial agent). Being rich entails that the information state contains a hierarchy of facts, including the ones that are thought to be shared and the ones that have not been yet publicised. We believe that ISU is a good basis for formalising existing theories of human-human dialogue (Cann et al., 2005; Ginzburg, 2012; Cooper, 2022) as a means to develop dialogue systems which produce acceptable and natural behaviours.

In controlled ISU-based systems, LLMs offer benefits such as interpreting user utterances and pre-generating dialogue domains (Larsson, 2024). Pre-generating a complete dialogue description (all possible dialogue paths) is a complex matter due to the inherent flexibility and variation in natural language dialogue. By taking care of a lot of this complexity using a flexible but controllable dialogue manager, the ISU approach considerably simplifies the pre-generation process. Instead of generating all possible dialogue paths, it is sufficient to generate a default dialogue "blueprint" which can then be used by rule-based dialogue manager as a resource when carrying out flexible and complex dialogue interactions.

At the same time, not all dialogue behaviours are necessarily best treated on an ISU level. This applies in particular to more routinised and "reflex" behaviours such as those involved in real-time incremental turntaking (Skantze, 2021; Howes et al., 2019, among others). Such behaviours are arguably better coded as statecharts.

So, how do we best combine ISU-based dialogue management, statecharts, and LLMs for flexible and controllable AI? We argue that this is best done by offering a single framework encompassing both ISU and statecharts for managing dialogue based on LLM-generated dialogue blueprints. We term our approach Statecharts-based implementation of Information State Update (SISU).[1]

The paper is organised as follows. Section 2 introduces the main considerations behind SISU. In Section 3 we describe the implementation of SISU, while Sections 4 and 5 provide an outlook on two main advantages of our framework: pre-generating

---

[1] Source code and online demo available at: `https://github.com/GU-CLASP/sisu`

Proceedings of the 2025 CLASP Conference on Language models And RePresentations (LARP), pages 28–37
September 8-9, 2025 ©2025 Association for Computational Linguistics

dialogue blueprints and coding low-level dialogue routines with statecharts. We provide brief conclusions in Section 6.

2 Method

This section introduces SISU, an ISU framework which combines the statecharts formalism (Harel, 1987) with modern features of the TypeScript programming language. SISU implements a version of IBiS1 (Larsson, 2002) and improves upon it by providing the capacity to code routinised dialogue phenomena using statecharts.

2.1 Information State Update

One of the central purposes of the ISU approach to dialogue management (Larsson and Traum, 2000; Larsson, 2002) is to enable the implementation and comparison of dialogue theories by casting them in a common form. Central questions in this endeavour are: (1) what kind of information does a dialogue participant need to keep track of, (2) how does this information get updated by utterances in dialogue and (3) how does this information license subsequent utterances?

In the ISU approach, utterances in conversation are seen as *dialogue moves* which trigger updates to a rich conversational Information State (IS), which then is used to select an appropriate followup dialogue moves. *Update rules* describe updates to the IS, and *selection rules* describe conditions on IS under which a move can be selected. For example, an information state can include a stack of questions that have been raised but not yet addressed. Given this, one update rule may state that asking a question Q results in pushing Q on the question stack, and a selection rule may state that if a question is on the question stack, and if an answer A to that question is known, an answer move with content A may be selected and uttered in the next turn.

An example of an ISU-based dialogue manager is Talkamatic Dialogue Manager (TDM) (Larsson and Berman, 2016) which is a core component of Talkamatic Studio[2], a web-based tool for generating, curating and deploying Pre-Generative Conversational AI agents. TDM handles a wide variety of dialogue behaviours and a wide range of dialogue types, including question-answering, search, device control, educational, instructional and negotiative dialogue. TDM uses Issue-Based Dialogue Management (Larsson, 2002) which is strongly influenced by KoS (Ginzburg, 2012).

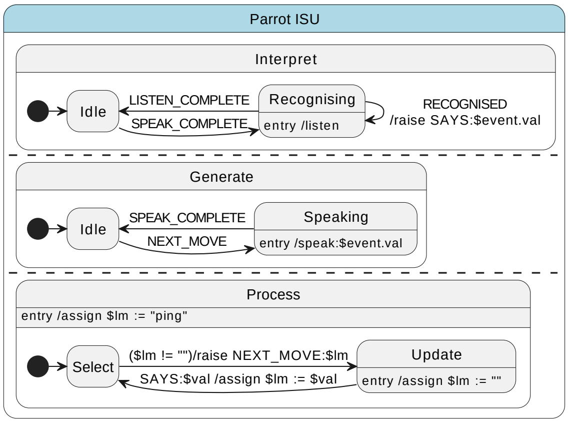

Figure 1: A statechart for "parrot" dialogue system. Events are typeset in all capitals, states – with initial capital. Conditions are parenthesised and actions are denoted by a starting "/" symbol.

2.2 Statecharts

Statecharts were originally developed by Harel (1987) for complex systems (real-time, multi-computer and concurrent). Harel introduces conventional notation over deterministic Finite-State Machines (FSMs), incorporating depth, orthogonality and broadcast communication. Statecharts can be represented graphically, as diagrams with a variable level of detail. A number of studies have demonstrated that statecharts can be useful for designing dialogue systems (Kronlid and Lager, 2007; Brusk, 2008; Mehlmann et al., 2011).

Figure 1 illustrates the use of three orthogonal (parallel) states in a simplest "parrot" dialogue system[3] implementing ISU with only one element of the IS – `$lm` (latest move). `Interpret` state implements speech recognition and can raise `SAYS` event with a recognised value; `Generate` state implements speech synthesis which speaks out the value of `NEXT_MOVE` event. The `Process` state implements dialogue management and is initiated with `$lm` containing an initial move "ping"; it starts in a `Select` state and, if `$lm` is not empty, raises a `NEXT_MOVE` event with a value of `$lm` and transitions to `Update` state. Then, upon receiving `SAYS` event it updates `$lm` with the event value and transitions back to `Select` state.

SISU is strongly inspired by Kronlid and Lager (2007) system, which used a version of State Chart

---

[2] `https://studio.talkamatic.se`

[3] Previously used by Bos et al. (2003); Kronlid and Lager (2007) to illustrate their implementations of ISU.

XML (SCXML) extended with Prolog-style conditions for state transitions. Our implementation uses XState[4], a state management library for JavaScript and TypeScript which implements Harel's formalism and uses actor model for concurrency. Despite an abundance of commercially available dialogue-building software based on FSMs and deterministic flows, such as Dialogflow[5], statecharts are supported by the World Wide Web Consortium (W3C) specification[6], which serves as a guidance for modern implementations, such as XState.

3 Architecture

In this section, we describe the general architecture of SISU, exemplified by a simple form-filling dialogue application in the context of scheduling.

3.1 Information state

For representing the information state, we utilise the data storage of XState, and declare the IS as a type:

```
type InformationState = {
  next_moves: Move[];
  domain: Domain;
  database: Database;
  private: {
    agenda: Action[];
    plan: Action[];
    bel: Proposition[] };
  shared: {
    lu: { speaker: Speaker; moves: Move[] };
    qud: Question[];
    com: Proposition[]; };};
```

The information state is composed of the following data structures:

moves Moves are essentially dialogue acts that can be performed by the user or the system. In SISU, they consist of a type (e.g., "answer" or "ask") and content.[7] For example, the type of the "ask" move is defined as follows:

```
type AskMove = { type: "ask";
                 content: Question; };
```

questions There are several types of questions that can be supported by the system, i.e. wh-questions which contain a predicate to be fulfilled.

```
type WhQuestion = { type: "whq";
                    predicate: string };
```

For instance, a question concerning the location of a booking can be represented as `{type: "whq", predicate: "booking_room"}`.

propositions A predicate combined with an argument forms a proposition. For instance, a proposition that a lecture takes place in G212 can be represented as `{predicate: "booking_room", argument: "G212"}`.

plans and actions Plans are domain-specific and high-level descriptions of how goals are achieved. Plans are represented as lists of actions. For example, a plan for responding to a question concerning the location of a booking can be represented as:

```
{ type: "issue",
  content: { type: "whq",
             predicate: "booking_room" },
  plan: [{ type: "findout",
           content: {
             type: "whq",
             predicate: "booking_course" }},
         { type: "findout",
           content: {
             type: "whq",
             predicate: "booking_day" }},
         { type: "consultDB",
           content: {
             type: "whq",
             predicate: "booking_room"}}]}
```

This plan enables the system to respond to questions concerning locations of a booking by first finding out which course and day the question concerns, and by then consulting a database.

3.2 Update rules

In SISU, update rules have the type

```
type Rule =
   (context: InformationState) =>
     ((x: void) => InformationState) | undefined;
```

which takes an information state as input and, depending on the truth value of the precondition, either, if the precondition is not met, returns `undefined`, or else a function that returns the updated state.[8] For instance, the following rule, which implements rule 2.2 from Larsson (2002), pushes a question recently uttered by the system, if such one exists, onto the stack of questions under discussion (QUDs).

---

[4] `https://stately.ai/docs/xstate`

[5] see Sabharwal and Agrawal (2020) and `https://cloud.google.com/dialogflow/cx/docs/basics`

[6] `https://www.w3.org/TR/scxml/`

[7] There is some similarity with the notion of intents, but those are typically domain-specific, whereas in our framework we assume that the update rules can be domain-general which requires operating over more abstract data structures.

[8] XState's `assign()` method is used when the rule is instantiated.

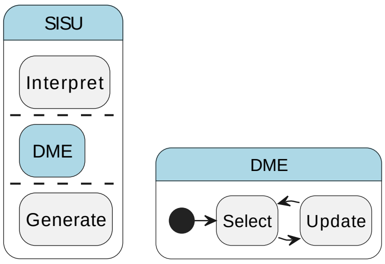

Figure 2: SISU and DME statecharts (zoomed-out).

```
integrate_sys_ask: ({ is }) => {
  if (is.shared.lu!.speaker === ”sys”) {
    for (const move of is.shared.lu!.moves) {
      if (move.type === ”ask”) {
        const q = move.content;
        return () => ({
          ...is, shared: {
            ...is.shared,
            qud: [q, ...is.shared.qud]
          }});}}}};
```

We make use of spread (...) JavaScript syntax, which effectively allows overriding parts of information state as well as operating stacks (i.e. pushing a question Q into the stack of QUDs).[9]

3.3 Dialogue Move Engine (DME)

Overall architecture of SISU (statechart on Figure 2) extends the "parrot" example with a Dialogue Move Engine (DME) statechart. It is responsible for updating the IS and selecting moves to be produced. The structure of DME reproduces the statechart introduced by Kronlid and Lager (2007).

Figures 3 and 4 show move selection and state update processes. We use syntactic sugar "(*/condition*)" to represent an update rule which consists of an update *action* performed in case when the *condition* is met. The update action and evaluator for the condition are provided as functions of the type introduced in Section 3.2. If no update rules can be applied in a given state, it transitions to the next one (depicted with arrows without conditions or event), except for the `Grounding` state in which DME waits for the SAYS event (containing either a recognised move or a move produced by the system) which triggers further update process. When the move selection is done, the move is taken up by the language generation state orthogonal to DME (similarly to Figure 1).

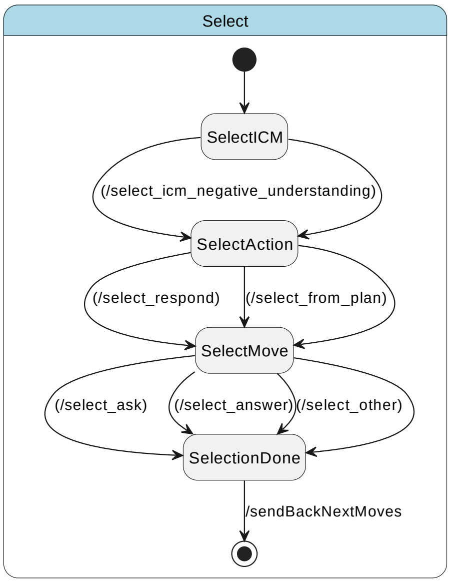

Figure 3: Zoomed-in `Select` statechart.

3.4 Domain knowledge and database

Following Larsson and Berman (2016), SISU separates general knowledge about dialogue from domain-specific knowledge. The developer (or a LLM, as we will show in Section 4) supplies the latter, e.g. dialogue plans and API integrations.

4 Pre-generating dialogue domains with LLMs

Despite their sometimes impressive performance, LLMs are associated with a host of well-known problems (hallucinations, bias etc.) deriving from the overall problem of controlling the behaviour of LLMs (Kann et al., 2022). The absolute majority of methods for dealing with this problem is of the "guardrails" type. In LLM-based Conversational AI, however, the user is still interaction with an LLM at runtime, and it is difficult or impossible to guarantee that guardrails always work. Ayyamperumal and Ge (2024) discuss various guardrail approaches such as layered protection models, system prompts, Retrieval-Augmented Generation (RAG) architectures and bias mitigation, and observe that "[c]rucial challenges remain in implementing these guardrails." Xu et al. (2024) show that hallucination is not just a temporary glitch, but are in fact inevitable in LLMs.

An alternative to using LLMs is of course to manually build dialogues, as done in e.g. Dialogflow (Sabharwal and Agrawal, 2020) and many other toolkits. In the SISU approach, we call such dialogue specifications *dialogue domain descriptions*,

---

[9] Ginzburg (2012); Cooper (2022) use asymmetric merge operation for expressing updates of an information state. For JavaScript object literals, when one object is spread into another, the property can be overridden by the last assigned value.

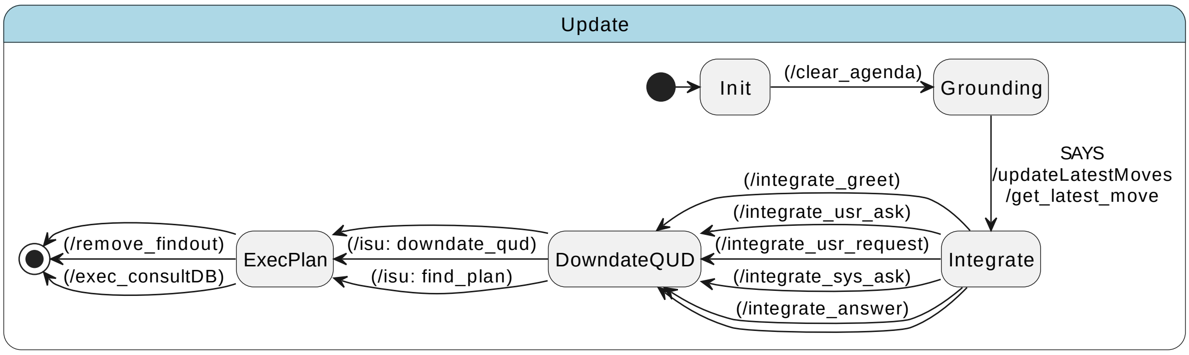

Figure 4: Zoomed-in `Update` statechart.

or dialogue domains for short. Although dialogue domains are concise and expressed on a high level of abstraction, coding them by hand can be time- and resource-intensive. Furthermore, the formalism can be challenging for non-technical domain experts. Pre-Generative Conversational AI (Larsson, 2024) uses LLMs for pre-generating domain knowledge to address these problems. Specifically, the LLM is fed relevant information such as input type definitions and example dialogues between system and user, and returns a dialogue domain and database integration. The interaction between an LLM and dialogue developer can be included in a workflow alongside user testing, enabling insights from testing to be fed to the LLM to inform subsequent improvements. We provide the details of our approach in Appendix A, namely, how an LLM can be used to generate domain and database for a small scheduling scenario.[10]

5 Coding low-level dialogue routines with statecharts

In addition to FSMs being a prominent framework for building dialogue systems (McTear, 2020), previous studies explained complex natural language grounding phenomena using FSMs, such as vocabulary enquiries in language tutoring (del Fresno et al., 2022) and spelling out names (Howes et al., 2019; Larsson et al., 2020). Such FSMs include routinised adjacencies between dialogue acts in human dialogue; to build a dialogue system based on these definitions one must include considerations from system perspective and cast FSMs into a statechart.[11] Figure 8 in Appendix B shows an example a statechart based on a FSM from Larsson et al. (2020). SISU can include routinised procedures defined in such way and involve ISU only when certain information is grounded, e.g., when the user and the system have demonstrated agreement on a particular spelling of a word.

6 Conclusions

In this paper we described a framework combining statecharts and ISU approach. We described its architecture and illustrated it with a small scheduling scenario. We underlined two main advantages of our approach, first, its adequacy for pre-generating dialogues with LLMs and, secondly, the possibility of mixing routinised dialogue procedures, such as task-specific grounding described with FSMs or statecharts, with ISU based on theoretical principles.

Future work will include empirical evaluation of the framework as well as expanding the case study to cover a range of more complex domains.

Acknowledgements

Vladislav Maraev was supported by Swedish Research Council (VR) grant 2023-00358 – Social laughter for virtual agents (SocLaVA). We also acknowledge support from the Swedish Research Council VR project 2014-39 for the establishment of the Centre for Linguistic Theory and Studies in Probability (CLASP) at the University of Gothenburg.

---

[10] An industry strength implementation of Pre-Generative Conversational AI is available in Talkamatic Studio. The examples in this paper are illustrative toy examples intended to convey the gist of the idea, and do not reflect how dialogue pre-generation is implemented in Talkamatic Studio.

[11] Statecharts can be also used for describing routines in human dialogue, because they allow less clutter thanks to state hierarchy.

References

Suriya Ganesh Ayyamperumal and Limin Ge. 2024. Current state of LLM risks and AI guardrails. *arXiv preprint arXiv:2406.12934*.

Johan Bos, Ewan Klein, Oliver Lemon, and Tetsushi Oka. 2003. DIPPER: Description and formalisation of an information-state update dialogue system architecture. In *Proceedings of the Fourth SIGdial Workshop of Discourse and Dialogue*, pages 115–124.

Jenny Brusk. 2008. Dialogue management for social game characters using statecharts. In *Proceedings of the 2008 International Conference on Advances in Computer Entertainment Technology*, ACE '08, page 219–222, New York, NY, USA. Association for Computing Machinery.

Ronnie Cann, Ruth Kempson, and Lutz Marten. 2005. *The Dynamics of Language*. Elsevier, Oxford.

Robin Cooper. 2022. *From Perception to Communication: a Theory of Types for Action and Meaning*. Oxford University Press, Oxford. In press.

Andrea Carrión del Fresno, Staffan Larsson, and Vladislav Maraev. 2022. Dialogue strategies for... cómo se dice entrenamiento de vocabulario? In *Proceedings of the 26th Workshop on the Semantics and Pragmatics of Dialogue - Full Papers*, Dublin, Ireland. SEMDIAL. 10 pages.

Jonathan Ginzburg. 2012. *The interactive stance*. Oxford University Press, Oxford.

David Harel. 1987. Statecharts: a visual formalism for complex systems. *Science of Computer Programming*, 8(3):231–274.

Christine Howes, Anastasia Bondarenko, and Staffan Larsson. 2019. Good call! Grounding in a Directory Enquiries Corpus. In *Proceedings of the 23rd Workshop on the Semantics and Pragmatics of Dialogue*, London, United Kingdom. SEMDIAL.

Katharina Kann, Abteen Ebrahimi, Joewie Koh, Shiran Dudy, and Alessandro Roncone. 2022. Open-domain dialogue generation: What we can do, cannot do, and should do next. In *Proceedings of the 4th Workshop on NLP for Conversational AI*. Association for Computational Linguistics.

Fredrik Kronlid and Torbjörn Lager. 2007. Implementing the information-state update approach to dialogue management in a slightly extended SCXML. In *Proceedings of the 11th International Workshop on the Semantics and Pragmatics of Dialogue (DECALOG)*, pages 99–106.

Staffan Larsson. 2002. *Issue-based dialogue management*. Department of Linguistics, Göteborg University.

Staffan Larsson. 2024. Pre-generative conversational AI. In *Proceedings of the 28th Workshop on the Semantics and Pragmatics of Dialogue - Poster Abstracts*, Trento, Italy. SEMDIAL.

Staffan Larsson and Alexander Berman. 2016. Domain-specific and general syntax and semantics in the talkamatic dialogue manager. *Empirical Issues in Syntax and Semantics*, 11:91–110.

Staffan Larsson, Christine Howes, and Anastasia Bondarenko. 2020. Could you spell that again please? Towards a formal model of grounding in directory enquiries. In *First AISB Symposium on Conversational AI (SoCAI)*.

Staffan Larsson and David Traum. 2000. Information state and dialogue management in the trindi dialogue move engine toolkit. *NLE Special Issue on Best Practice in Spoken Language Dialogue Systems Engineering*, pages 323–340.

Michael McTear. 2020. *Conversational AI: Dialogue systems, conversational agents, and chatbots*, volume 13. Morgan & Claypool Publishers.

Gregor Mehlmann, Birgit Endraß, and Elisabeth André. 2011. Modeling parallel state charts for multithreaded multimodal dialogues. In *Proceedings of the 13th International Conference on Multimodal Interfaces*, ICMI '11, page 385–392, New York, NY, USA. Association for Computing Machinery.

Navin Sabharwal and Amit Agrawal. 2020. Introduction to Google Dialogflow. In *Cognitive virtual assistants using google dialogflow: develop complex cognitive bots using the Google Dialogflow platform*, pages 13–54. Springer.

Gabriel Skantze. 2021. Turn-taking in conversational systems and human-robot interaction: A review. *Computer Speech & Language*, 67:101–178.

Ziwei Xu, Sanjay Jain, and Mohan Kankanhalli. 2024. Hallucination is inevitable: An innate limitation of large language models. *arXiv preprint arXiv:2401.11817*.

A Pre-generating a dialogue domain with ChatGPT

In Pre-generative Conversational AI (Larsson, 2024) implemented on industrial level in Talkamatic Studio, dialogue domain data is generated by an LLM from content such as a database API or a text. The TDM uses this data to enable flexible dialogue in various types of dialogue, including educational, customer service, instructional and negotiative dialogue.

Here, as a simple demonstration of the feasibility of pre-generating dialogue domains using LLMs, we feed a dialogue example to ChatGPT (`GPT-4o`), together with the type definition for a domain and database, and ask it to generate a domain and database that supports the dialogue (see Figure

5). The model's generated domain and database are included in Figures 6 and 7 respectively[12].

Automated testing of the system's resulting behaviour validates that the model's generated code works as intended, without any errors. It is worth noting that except for a minimal amount of comments in the type definition, no documentation of the dialogue domain formalism or the overall dialogue system is provided.

One can also note that the code generated by an LLM contains a semantic peculiarity: all predicates and individuals are declared as a single sort (`course`). This can seem unintuitive, since days and locations are not courses. One consequence of this peculiarity is that the answer "Thursday" will be considered relevant in relation to the question "Which course?". This problem can potentially be addressed within an LLM-based development framework by extending the set of dialogue examples fed to the LLM.

B Coding parts of dialogue with statecharts

Figure 8 shows an example of how procedural FSM for name-spelling derived from human-human dialogue (Howes et al., 2019; Larsson et al., 2020) can be adapted to statechart definition for further use in SISU-based dialogue system.

---

[12] Note that this is an illustrative toy example and does not reflect how dialogue pre-generation is implemented in Talkamatic Studio.

```
Create a domain and database that supports the dialogue below between user
(U) and system(S), and that adheres to the type definition below.

SUPPORTED DIALOGUE

S: Hello! You can ask me anything!
U: Where is the lecture?
S: Which course?
U: Dialogue Systems 2
S: Which day?
U: Thursday
S: The lecture is in G212.

TYPE DEFINITION

type Individuals = Predicates;
type Predicates = { [index: string]: string };
export type Domain = {
  plans: PlanInfo[];
  predicates: Predicates;  // Mapping from predicate to sort
  individuals: Individuals; // Mapping from individual to sort
};

export type PlanInfo = {
  type: ”action” | ”issue”;
  content: null | Proposition | ShortAnswer | Question;
  plan: Action[];
};

export type Database = {
  consultDB: (q: Question, p: Proposition[]) => Proposition | null;
};

export type ShortAnswer = string;
export type Proposition = {
  predicate: string;
  argument: string;
};

export type Question = WhQuestion;
type WhQuestion = { type: ”whq”; predicate: string };

export type Action = {
  type:
    | ”greet”
    | ”respond” // not to be used in plans
    | ”raise”
    | ”findout”
    | ”consultDB”;
  content: null | Question;
}
```

Figure 5: Input fed to ChatGPT.

```
export const domain: Domain = {
  predicates: {
    course_day: "course",
    course_location: "course",
    course_name: "course",
  },
  individuals: {
    "Dialogue Systems 2": "course",
    "Thursday": "course",
    "G212": "course",
  },
  plans: [
    {
      type: "issue",
      content: { type: "whq", predicate: "course_location" },
      plan: [
        { type: "findout", content: { type: "whq", predicate: "course_name" } },
        { type: "findout", content: { type: "whq", predicate: "course_day" } },
        { type: "consultDB", content: { type: "whq", predicate: "course_location" } }
      ]
    }
  ]
};
```

Figure 6: Domain generated by ChatGPT.

```
export const database: Database = {
  consultDB: (q, props) => {
    if (
      q.type === "whq" &&
      q.predicate === "course_location" &&
      props.some(p => p.predicate === "course_name" && p.argument === "Dialogue Systems 2") &&
      props.some(p => p.predicate === "course_day" && p.argument === "Thursday")
    ) {
      return { predicate: "course_location", argument: "G212" };
    }
    return null;
  }
};
```

Figure 7: Database generated by ChatGPT.

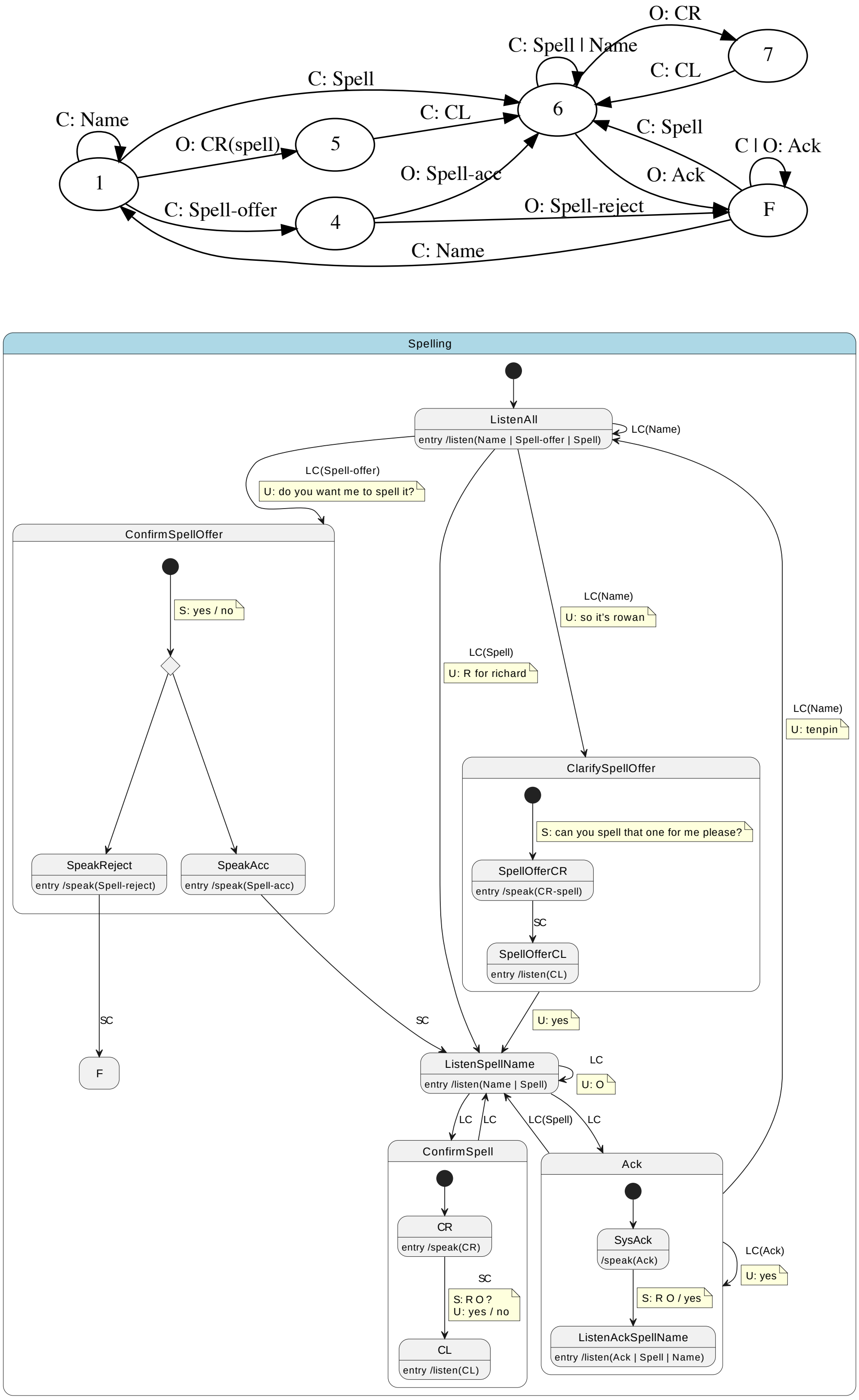

Figure 8: **Above**: name-spelling FSM adapted from Larsson et al. (2020). **Below**: a corresponding statechart. SC = system yields the turn; LC = user yields the turn. Yellow notes exemplify possible utterances.